

**APPLICATION  
FOR  
UNITED STATES LETTERS PATENT**

**TITLE:            SECURE DATA EXCHANGE BETWEEN DATA PROCESSING  
SYSTEMS**

**APPLICANT: Larry T. Harada  
Mark A. Dolecki,  
Christopher S. Purdum  
C. Hudson Hendren III**

"EXPRESS MAIL" Mailing Label Number EE6472792/305

Date of Deposit

June 1, 1999

I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as "Express Mail Post Office To Addressee" with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Francisco Robles

Francisco Robles

## SECURE DATA EXCHANGE BETWEEN DATA PROCESSING SYSTEMS

### BACKGROUND

Fig. 1 shows a computer network 100 that includes both client computers 111-113  
5 and server computers 131-133. The network 100 also includes data connections that allow  
the client computers to communicate with the servers. Those data connections include access  
connections 114-116, a service provider's point of presence (POP) 110, network 120, proxy  
server 117, and network 130. POP 110 includes data communications equipment that allows  
a client computer 111-113 to connect to the network 120. For example, POP 110 may include  
10 dial-up modem banks, cable modem banks, a T1 interface, wireless communications  
equipment, or other data transmission equipment.

After a client computer 111-113 has been connected to the network 120 (through POP  
110) it may exchange data with other computers on the network 120. Computers on different  
networks 120 and 130 also can exchange data with each other by sending their data request to  
15 proxy server 117 which, in turn, can forward their request to the destination computer, obtain  
a response from the destination, and return the response to the requesting computer.

The servers 131-133 and clients 111-113 may communicate with each other using the  
hypertext transfer protocol (HTTP). HTTP is a data communications protocol that is used by  
web browser programs (such as Microsoft Internet Explorer and Netscape Navigator) to  
20 communicate with web server applications (such as Microsoft Internet Information Server  
and Netscape Server). Thus, by executing a web browser, a client computer 111-113 can  
exchange data with a server 131-133 that is executing a web server application.

When a web server receives a HTTP data request from a browser, a web server  
application, such as a Common Gateway Interface (CGI) script, may be used to query  
25 databases, customize responses, and/or perform other processing used to generate a response.  
The web server application may need data about the client computer, the browser, its user,  
and/or other data in order to complete its processing of the data request. For example, a web  
server application may provide customized weather information based on a user's home  
address. The web application may obtain the user's address information by sending a data

input form to the browser and receiving a user's response that contains the needed information. A web server application also may obtain needed data from web server environmental variables that are populated by data in HTTP header fields, by a web server, and/or by a web server sub-component. For example, a CGI script's response to a data request may depend on the type of browser originating the request. The CGI script may access a HTTP environment variable ("HTTP\_USER\_AGENT") populated by the web server using data from a "User-Agent" field in a HTTP header. The value of the HTTP\_USER\_AGENT variable identifies the type of browser originating a request so that the CGI script can customize its response to that browser.

A web server and a browser also may exchange web cookies that contain data. A web cookie is data file that can be automatically exchanged between browsers and servers along with the request and responses exchanged between the browsers and servers. When a web cookie is received by a browser, it can be stored at the client computer. During future transactions with the web server, the web cookie (and any data that it contains) can be sent back to the web server. By using data stored in web cookies, a web server's need to request data from a user may be reduced. However, since web cookies are typically stored on a client computer, they may be deleted and their data lost. Furthermore, if a user moves to another computer, web cookie information stored on the user's original computer will be unavailable to that user. These factors may reduce the utility of web cookies.

The present inventors recognize that information service systems, such as web servers, should provide convenient, secure, reliable, and simple means of interacting with users. The present inventors further recognize when a server must query a user for data or use web cookies for data storage and input, convenience, security, reliability, and simplicity may suffer. Consequently, the invention disclosed herein provides additional means of providing information to a web server application that may provide additional flexibility in implementing information systems.

## SUMMARY

In general, in one aspect, the invention features a data transfer method performed at a proxy server. The method includes intercepting a data request from a client computer that is directed to a target server, encrypting profile information, augmenting the data request by

adding the encrypted profile information to the data request, and sending the augmented data request to the target server. In another aspect, the invention features a data transfer method performed at an information server. The method includes receiving a data request from a proxy server, extracting profile information added to the data request by the proxy server, using the extracted profile information to generate a response, and sending the response to the proxy server.

In general, in another aspect, the invention features a computer program residing on a computer-readable medium. The program includes instructions for causing a computer to intercept a data request from a client computer that is directed to a target server, encrypt profile information, augment the data request by adding the encrypted profile information to the data request, and send the augmented data request to the target server. In another aspect, the invention features a computer program residing on a computer-readable medium that includes instructions for causing a computer to receive a data request comprising encrypted profile information that was added to the data request by a proxy server, extract the profile information added by the proxy server, use the extracted profile information to generate a response, and send the response to the proxy server.

In general, in another aspect, the invention features a proxy server that includes a database, a network interface, a processor, and a memory. The database includes records storing user profile information. The network interface operatively couples the proxy server to a network to exchange data with a client computer and with a target server. The processor is operatively coupled to the network interface, the database, and a memory. The memory includes executable instructions for causing the processor to intercept a data request that is directed to a target server, retrieve a record from the database, encrypt profile information in the record, augment the data request by adding the encrypted profile information, and send the augmented data request to the target server.

~~In general, in another aspect, the invention features an information server that includes a network interface, a processor, and a memory. The network interface operatively couples the information server to a proxy server. The processor is operatively coupled to the network interface and to the memory. The memory includes executable instructions for causing the processor to receive a data request from the proxy server, decrypt user profile~~

information added to the data request by the target server; and use the decrypted user profile information to generate a response to the data request.

Implementations may include one or more of the following features. A reference token may be exchanged between servers and used to refer to previously exchanged profile information. The reference token may be placed in subsequent data request sent by the proxy server to the target in place of full profile information. The profile information that is encrypted by the proxy server can be stored in proxy server database records. The particular user profile information encrypted and included in a data request can be selected based on an identity of a client computer user or a browser user. The identity may be determined using the Internet Engineering Task Force IDENT protocol or by using another method.

Implementations also may include one or more of the following features. Profile information may be encrypted at the proxy server using a session key as an encryption key, and may be decrypted at the target server using the same session key. The session key may be determined by the proxy server and sent to the target server using a public key cryptography algorithm and a public key associated with the target server. A data request may use the hypertext transfer protocol. Profile information extracted from the data request at the target / information server can be provided to a web application, such as a CGI script, that is used to generate a response.

The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Implementations may provide advantages including reduction of redundant data entry, reduction of user data entry errors, secure storage of user data, and automated user profile determination and dissemination. Different encryption keys can be used for different information servers, and each information server may establish characteristics associated with its encryption key. A web site may dynamically generate, maintain, and change its unique encryption key. This may help to minimize risk of replay attacks, message forgery, and message tampering. In addition, implementations may support multiple levels of security.

Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

## DESCRIPTION OF DRAWINGS

FIG. 1 is a computer network diagram.

FIG. 2 is a computer network diagram.

FIGS. 3A and 3B show exemplary HTTP headers.

5 FIGS. 4A and 4B are flowcharts.

FIG. 5 is a message flow diagram.

## DETAILED DESCRIPTION

Processing of an information request at a information server, such as a web server, can be facilitated by a data transfer mechanism in which data needed to process the request is stored at a proxy server and automatically transferred to the information servers. Such a data transfer mechanism can be used, for example, by an information service provider (ISP) or on-line service provider (OSP) to automatically transfer information about a user (a “user profile”) to affiliated web sites. Referring to Fig. 2, to automatically transfer a user profile to a web server 203, the user profile information is first stored at a proxy server 202 in a database 220. When a HTTP data request 211 is received by the proxy server 202 from a web browser 201, the user profile information from database 220 is encapsulated in a request 212 that is forwarded by the proxy server to web server 203. The proxy server 202 encapsulates the user profile information in the request 212 by adding HTTP headers containing the user profile information to the headers received in the request 211.

20 Figs. 3A and 3B shows fields in HTTP requests. The fields 300 may be fields in a request 211 from a client computer 201, and the fields 350 may be fields in a request 212 forwarded by proxy server 202 to a web site 203. Referring to Figs. 3A and 3B, each field 301-303 includes a HTTP field name and a value associated with the field. HTTP field names identify the associated field value. For example, the “User-Agent:” field name in field 302 indicates that request 300 originated at a “Mozilla/3.0 compatible” browser; and the “Cookie:” field name in field 303 includes the value “been\_here\_before” is a web cookie. When a HTTP request 211 having fields 300 is received at proxy 202, the proxy server 202 can add user profile information to the request 211 by adding additional HTTP fields 304-305 containing the profile information, thereby generating a new request 212 with the user profile information encapsulated in fields 304-305.

Database 220 may be a local or remote database that can store multiple user profiles. For example, the database 220 may be a database storing user profiles on a hard disk directly connected to the proxy server 202, or the database may be a SQL database at a remote system and accessed over a TCP/IP connection. Still other database 220 implementations may be used. Each user profile may be associated with a particular user or group of users. A user profile may be selected from the database 220 based on the identifying information associated with a particular computer or user of that computer. To determine the user's identity, a proxy server may use a table or database that associates user identity information with network connection information. For example, referring to Figs. 2, 4A, and 4B, when a client computer 201 connects to a network 230, the client computer's user may submit name and password information to a POP or to a login server. The POP or login server may then send the user name and network connection information unique to that user (such as a unique combination of TCP/IP address and port number) to the proxy server 202 where it is stored in a database 220 (step 400). When the proxy server 202 receives a subsequent HTTP request 211 (step 401), the proxy server can identify the user associated with the request by querying the database of stored name and network connection associations. A proxy server also may identify a user using the IDENT protocol. The IDENT protocol is an Internet protocol that allows a computer to ask another computer for the name of the user. The IDENT protocol is further described in Internet Engineering Task Force (IETF) document RFC1413.

When the proxy server 202 has identified the user (step 402), the proxy server can retrieve a user profile associated with that user from its database 220. The proxy server may then add HTTP fields containing user profile information to fields in the original HTTP request 211 (step 403). For example, header fields 304-305 may be added to the original header fields 301-303 from request 211. Field 304 may include unencrypted user profile data ("UserName=John\_Doe, ZipCode=60609, ParentalControl=YoungTeen), while field 305 may include encrypted user profile data. Encryption of data in the field 305 may use the SecureData protocol (explained later). The proxy server thereby forms the modified HTTP request 212 containing the user profile data. The modified request 212 may then be forwarded by the proxy server for receipt at a web site (steps 404 and 421).

Particular web servers may or may not be configured to recognize the user profile data fields 304-305. A web server that is not configured to recognize the user profile fields 304-305 may ignore the fields. A web server that is configured to recognize the user profile fields 304-305 can extract the profile data from the field 304-305 (steps 422 and 424).

5 Extraction of user profile data in the field 305 may include decryption of the user profile data in encrypted field 305. The extracted user profile data can be used, for example, to generate or customize data sent in response 213 to the proxy server for forwarding as a response 214 to a user or client computer (step 425). For example, a tourist information web server may customize a page based user profile data specifying a browser user's age, and interest. If a  
10 web server looks for a user profile field 304-305 in the request 212, and the user profile field is not present, the server may use "traditional" data input mechanisms (such as forms and web cookies) to obtain needed data from a user (step 422-423). After the necessary data is obtained, the web server 203 can generate a response (step 425).

Exchange of user profile data between a proxy server 202 and a web server 203 may  
15 implemented using custom filtering software ("proxy data exchange filter" software) that can be added to commercially available proxy and web site server software. For example, Microsoft Internet Information Server (IIS) provides an application programming interface known as the information server application programming interface (ISAPI) that allows custom filtering software to be added to an IIS web site. A Netscape web site server provides  
20 an application programming interface known as the Netscape server application programming interface (NSAPI) that may also be used to develop custom filtering software. When a request 212 from the proxy server 203 is received at an IIS or Netscape server web site, the request 212 is passed to the proxy data exchange filter by the IIS or Netscape server software and the user profile data is extracted from the headers 304-305. The extracted  
25 profile data may then be made available as HTTP environment variables, database entries, or other data exchange structures that can be used by web applications. The remaining header information in request 212 (e.g., the original information from request 211), may then be processed by the web server to obtain response 213 information.

A proxy server 202 and web server 203 may implement the SecureData protocol  
30 (described below) to encrypt user profile data. Encrypted user profile data may be indicated



by a special HTTP header. For example, the field name "User-Profile-SecureData" in field 305 indicates that the field value segment of field 305 contains encrypted profile data.

The SecureData protocol uses multiple cryptography algorithms to provide secure user profile data transfers between a proxy server 202 and a web server 203. For example, the SecureData protocol may use the Rivest, Shamir, Adelman (RSA) public key encryption algorithm or the pretty-good-privacy (PGP) public key cryptography algorithm along with the RSA RC4 symmetric encryption algorithm. Additional information on the RC4 algorithm can be found, for example, in *Applied Cryptography*, 2nd edition, By Bruce Schneier, John Wiley & Sons, 1996.

User profile information may be encrypted at proxy server 202 using a symmetric encryption algorithm. A symmetric encryption algorithm, such as the RSA RC4 algorithm, uses the same encryption key (referred to herein as a "session" key) to encrypt and decrypt data. The proxy server 202 may encrypt user profile data and place it in a HTTP field in request message 212. The proxy server 202 can also separately encrypt the session key using a public key cryptography algorithm and the public key of the web server 203. The public key encrypted session key may also be placed in the request message 212, or may be sent separately to the web server 203. When the web server 203 has received the encrypted session key and user profile data, the web server 203 can decrypt its session key by using the public key cryptography algorithm and the web server's private key. The web server 203 may then decrypt the user profile information using the decrypted session key. Fig. 5 shows further details of an implementation in which a request message may include both the encrypted session key and the encrypted user profile data.

Referring to Fig. 5, when a request 511 is received at the proxy server 502, the proxy server determines whether it should add encrypted user profile data to the request. To do so, the proxy server may extract a destination web site address from URL information in the request 511, and query a database to determine whether the destination web site is to receive user profile information and whether the information is to be encrypted. If the destination web site is to receive encrypted user profile information, the proxy server 502 may then query its database to determine whether it has a valid public key for the web server 503. If the proxy server 502 has a valid public key, the proxy server can use the public key to

encrypt session key information that can be provided to a symmetric encryption algorithm to encrypt user profile data. The encrypted session key and user profile data may then be forwarded in requests 514 to the web server 503.

If the proxy server 502 does not have a valid public key for web server 503, the proxy server will obtain the public key from the web server 503 using a HTTP request 512. The request 512 includes a URL that identifies a public key file ("pub\_key\_file") stored on the server 503. The public key file may include additional information used by the proxy server 502. For example, the public key file may include the length of the public key, a SecureData protocol version number, a list of supported encryption algorithms, and shortcut configuration information ("shortcut" information is further explained, below).

An exemplary public key may have the format shown below (chevrons '<' and '>' delineate fields in the public key file and brackets '[' and ']' delineate optional information):

#### Exemplary Public Key File

```
<public key = length-of-key key>  
[<Version = SecureData protocol version number>]  
[<crypto = <encryption method>]  
[<shortcut = [none|low|med|high|max]>]
```

The web server's public key file is returned to the proxy server 503 in a HTTP response 513. The web server 503 may provide additional SecureData protocol information in HTTP fields of response 513 and/or in the public key file returned by the response message 513. For example, HTTP fields in response 513 may specify a time to live (TTL) for the key. The TTL value can be used to indicate the period during which the public key is valid.

A proxy server 502 or web server 503 may check a SecureData protocol version number returned in the public key file, in request 512 and/or in response 513 to ensure that the protocol is functioning properly and to recover from or to avoid errors. For example, if a proxy server 502 detects that protocol versions numbers in a retrieved public key file and those in a header of a response 513 do not match, the proxy server may re-submit the request 512 to the server 503. When the request 512 is re-submitted, the proxy server 502 can include the HTTP "no-cache" pragma to instruct caching systems between the proxy 502 and the web server 503 not to use cached data. This may help avoid errors introduced by using stale

cached data. When the proxy server 502 has received the web server's public key, it may cache the public key subject to a time-to-live (TTL) value specified by the web server 503 and/or the proxy server's own TTL policies.

If the proxy server 502 is unable to retrieve a web server's public key information or the public key information is otherwise unusable, the proxy server can add a problem report information to fields in a HTTP request 514 to indicate why the public key information is unusable. For example, the proxy server may indicate that the key information has been corrupted or a protocol version number or encryption level specified in the public key file is not supported. If the a web server's public key information is unusable and, consequently, the SecureData encryption protocol cannot be used, a proxy server may continue to forward the HTTP requests from browser clients to the web server without adding additional SecureData proxy headers. Alternatively the proxy server may block or otherwise filter request to the web server. The specific behavior may be defined on a per-web server basis using configuration information stored at the proxy server 502.

Once the proxy server 502 has the public key of web server 503, the proxy server can use the public key to encrypt session key data. The encrypted session key data may then be added to HTTP fields in the request from a browser or client computer 511 to form a modified request 514 that includes the public-key-encrypted session key information. The proxy server also can encrypt the user profile data using the session key information and a symmetric encryption algorithm and place the encrypted user profile data in HTTP fields in the modified request 514. For example, referring to Fig. 3B, the field 305 includes session key data "session=a3f792b210dafad" that may be public-key encrypted, and user profile data "Profile=A389C2ZA845MRAS02VMA39v1Z93AYC39AC3ABCEG78BB" that may be encrypted using a symmetric algorithm.

Different web server's or groups of web servers can have different session keys. A web server's session key may be formed by combining a "master" session key that is the same for a group of web servers with a key mask that can be used to create a unique session key for a web server or a group of web servers. For example, a 64-bit master session key may be combined with a 64-bit key mask using an exclusive-or ('XOR') operation, to yield a unique 64-bit session key. In some implementations, the master session key may be the proxy

server's public key. Other methods of generating unique keys can be used. The session key data can in field 305 (e.g., "session=a3f792b210dafad") can include numerous sub-fields. For example, the session key data can include a master key, a key mask, a timestamp, a URL hash, a Proxy IP field, and an encoding scheme field. These fields can contain the following data:

**Master key:** The master key is a value that is generated by the SecureData proxy server that can be combined with a key mask to generate a unique session key. The proxy server may periodically generate a new master session key. For example, a new master session key may be generated every  $n$  minutes. The Master key may be the proxy server's public key.

**Key mask:** The key mask is a value generated by a SecureData proxy server that is unique to a particular web site and which can be combined with a master key to generate a unique session key for that web site. A master key and a key mask can be combined using an exclusive-or ('XOR') operation, by appending the keys together, or by another key combination algorithm. A new key mask may be periodically generated. For example, a new key mask may be generated every  $n$  minutes.

**Timestamp:** A timestamp value may be added to the public-key encrypted data to make replay attacks more difficult. The timestamp value can be checked by the receiving web server and compared to the current time. If the difference between the timestamp value and current time exceeds a threshold value, the request may be denied or other error recovery or security procedures may be employed.

**URL hash:** A hash value derived from a requested URL may be added to the public-key encrypted data to make forging or tampering more difficult.

**Proxy IP:** The IP address of the proxy server be inserted in the session key data. When a request 514 is received at a web server 503, the web server may compare the IP address in the proxy IP field to help confirm that the request came from a particular proxy server 502. If the addresses differ, the request 514 may be rejected or other error recovery or security procedures may be employed.

**Encoding scheme:** Encoding scheme data may be sent from the SecureData proxy server to a SecureData web server to identify how user profile data is encoded. For example, the

encoding scheme value may indicate that profile data is in a binary (bin) form, or in name-value pair (nvp) form (which may be expressed using the Abstract Syntax Notation 1 (ASN.1) or other name-value coding scheme), or in another form.

The proxy server 502 can use a web server's unique session key to encrypt user profile data that is add it to the field 305. In some implementations, the user profile data may be placed in one or more additional fields. The request 514, now containing the public-key encrypted session data and the symmetric encrypted user profile data, is then sent to web server 503.

*Sub a3*  
10 ~~When request 514 is received by the web server 503, the request can be passed to proxy data exchange filter software that can extract the added fields 304-205 from the request 514, decrypt the session key and the user profile information contained in the fields 304-205, and make the user profile information available to web server applications. The user profile information may be made available to web server applications by setting HTTP environment variables, by storing it in a database, by placing it in shared memory, and/or using other data exchange techniques. The proxy data exchange filter software may then pass the request 514~~  
15 ~~back to the web server for further processing and for generation of a response 515.~~

The web server 503 can store received profile data in a local database for future request processing purposes. The web server 503 may return a shortcut token to the proxy server 502. The proxy server 502 may add the shortcut token to a subsequent request 518 in place of the 'full' user profile data sent in the request 514. This can be used to reduce the amount of data that needs be transferred in subsequent request 518. A shortcut token can be an index value, database query information, file name, other pointer data, or an arbitrary value generated by the web site 503 and used to reference the stored user profile data.

A web server's public key file may be sent to the proxy server 502 using a security level specified by the site 503. For example, a web site's public key file include data indicating that the web site will use a specified one of the following shortcut key security levels:

**none:** If "none" is specified, the shortcut token is sent to the proxy server as a clear (e.g., unencrypted value).

**low:** If “low” is specified, the shortcut token is sent to the proxy server as a clear value and is accompanied by a hash of a previously requested URL.

**medium:** If “medium” is specified, the “low” security level information is further accompanied by a timestamp value.

5 **high:** If “high” is specified, the “medium” security information is further accompanied by the proxy server's IP address.

**max:** If “max” is specified, the “high” security information is further accompanied by a sequence counter value. The sequence counter value may be based on a initial URL hash, the key mask, or other value, and is incremented in  
10 subsequent transactions.

At low, medium, high, and max security levels, the shortcut token and related security data can be sent in encrypted form using, for example, the proxy server's public key (which may be the master key value). Shortcut token security levels may also be indicated using HTTP fields in a response 513 or 515, or may be specified using data stored in a  
15 configuration database at the proxy server 513 using a data entry terminal.

A proxy server 502 or web server 503 may include other HTTP fields in a request 512, 514, 518 or in a response 513, 515, 519 to control user profile data transfers and/or the operation of the SecureData protocol. For example, HTTP fields in requests and/or responses may specify the following directives:

20 **Flush = <shortcut value | ALL>:** The ‘Flush’ directive can be sent to the proxy server 502 to discard a specified shortcut token or to discard all shortcut tokens from a particular web server. The ‘Flush’ directive may also be sent to a web server to cause the web server to discard stored user profile data associated with a particular shortcut token or to discard all stored user profile data associated with the sending proxy server. This  
25 directive may be used by a web server 503 or proxy server 502 that is performing a reset operation.

**WrongKey:** The WrongKey directive indicate that improper security protocol information was received. A proxy server that receives the WrongKey directive may re-send a request using a HTTP “no-cache” pragma. The “no-cache” pragma indicates that  
30 cached data should not be used to satisfy the request.

**SendFull:** A web server can issue a SendFull directive to indicate that a proxy server should send "full" user profile data (rather than a shortcut token) in a subsequent HTTP request.

**TTL = <n>:** The TTL (time to live) directive specifies a period (n, in seconds) during which a shortcut token is valid. Similar directives may be included in requests or response to indicate time-to-live values for other data, such as session keys and public keys.

The invention may be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention may be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps of the invention may be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on input data and generating output. The invention may advantageously be implemented in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program may be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language may be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing may be supplemented by, or incorporated in, specially-designed ASICs (application-specific integrated circuits).

A number of embodiments of the present invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, server, proxy, and client

protocols need not use the HTTP protocol. Alternate protocols and data formats may be used such as file transfer protocol (FTP) or network news transfer protocol (NNTP). Accordingly, other embodiments are within the scope of the following claims.